

DOCKET No.  
NAI1P100/01.314.01

U.S. PATENT APPLICATION  
FOR A  
SYSTEM, METHOD AND COMPUTER PROGRAM  
PRODUCT FOR TRANSLATING SNMP (ASN.1)  
PROTOCOL DECODES

ASSIGNEE: NETWORKS ASSOCIATES TECHNOLOGY, INC.

1009061.030102

SILICON VALLEY IP GROUP  
P.O. Box 721120  
SAN JOSE, CA 95172

# SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR TRANSLATING SNMP (ASN.1) PROTOCOL DECODES

5

## FIELD OF THE INVENTION

The present invention relates to network analyzers, and more particularly to decoding network communications utilizing a network analyzer.

10

## BACKGROUND OF THE INVENTION

Network assessment tools referred to as "analyzers" are often relied upon to analyze networks communications at a plurality of layers. One example of such  
15 analyzers is the Sniffer® device manufactured by Network Associates®, Inc. Analyzers have similar objectives such as determining why network performance is slow, understanding the specifics about excessive traffic, and/or gaining visibility into various parts of the network.

20 In use, network analyzers often take the form of a program that monitors and analyzes network traffic, detecting bottlenecks and problems. Using this information, a network manager can keep traffic flowing efficiently. A network analyzer can also be used legitimately or illegitimately to capture data being transmitted on a network. For example, a network router reads every packet of data passed to it, determining  
25 whether it is intended for a destination within the router's network or whether it should be passed further along the Internet. A router with a network analyzer, however, may be able to read the data in the packet as well as the source and destination addresses. It should be noted that network analyzers may also analyze

data other than network traffic. For example, a database could be analyzed for certain kinds of duplication, etc.

Prior Art Figure 1A illustrates an exemplary architecture 10 showing the use  
5 of a network analyzer. In particular, the present example shows the use of a network analyzer in the context of a network using a simple network management protocol (SNMP). As shown, at least one console 12 communicates with a plurality of agents 14 using SNMP.

10 Coupled to at least one network segment between the console 12 and the agents 14 is a network analyzer 16. In addition to the various network analyzer functionality set forth hereinabove, one particular use of such network analyzer 16 is to decode frames that are communicated between the console 12 and the agents 14 for troubleshooting, etc. Decoding is a well known technique used by network  
15 analyzers for understanding the frames of communication.

As frames are decoded by the network analyzer 16 data is stored in a buffer "object." In the context of the present description, an object may refer to a buffer, memory, a table or any other set of data that is associated with a specific component  
20 of a communication protocol. Often, such objects include a hierarchical tree structure. Prior Art Figure 1B illustrates an exemplary hierarchical tree structure 20 of objects 22.

In order to accomplish this decoding, the network analyzer 16 is equipped  
25 with access to at least one management information base (MIB) 18. MIBs 18 are well known data structures that are traditionally compiled in order to generate software programs used by the network analyzer 16 to decode particular objects. Conventionally, different MIBs 18 are provided for decoding different objects.

Prior Art Figure 1C illustrates a graphical user interface 30 showing a plurality of objects 32 that are displayed as a result of a decode. As shown, associated with each of such objects 32 are numerical identifiers 34 which identify each of the objects resulting from the decoding. Unfortunately, it is difficult to  
5 analyze network traffic represented by such objects using the numerical identifiers 34, since they do not provide any intuitive information. In fact, such numerical identifiers 34 must often be manually deciphered in order to gain a true understanding of the decoded frames.

10 There is thus a need for a technique of gaining an automatic, intuitive understanding of decoded objects outputted as a result of a network analysis.

#0090264.030102

**DISCLOSURE OF THE INVENTION**

5 A system, method and computer program product are provided for translating  
protocol decode objects. Initially, a plurality of frames is received. Next, the frames  
are decoded in order to generate protocol decode objects each with a numerical  
identifier associated therewith. Still yet, the numerical identifier is translated to a  
textual identifier. The textual identifier associated with the protocol decode objects  
are then displayed for facilitating the use of the protocol decode objects during  
10 network analysis.

In one embodiment, the protocol may include SNMP (ASN.1). Further, the  
numerical identifier is translated to a textual identifier utilizing a map. To generate  
such map, a list of management information bases (MIBs) is initially received from a  
15 user. Such MIBs may include a hierarchical structure. Next, the list of MIBs is  
compiled in order to generate a map. Further, a decoder is loaded with the map so  
that the numerical identifier may be translated to the textual identifier utilizing the  
map during the decoding.

20 As an option, the map may include a look-up table. In particular, the map  
may include a list of the numerical identifiers each with an associated textual  
identifier. Further, the textual identifier may include alphanumeric text descriptive of  
the protocol decode objects.

25

**BRIEF DESCRIPTION OF THE DRAWINGS**

5 Prior Art Figure **1A** illustrates an exemplary architecture showing the use of a network analyzer.

Prior Art Figure **1B** illustrates an exemplary hierarchical tree structure of objects.

10 Prior Art Figure **1C** illustrates a graphical user interface showing a plurality of objects that are displayed as a result of a decode.

Figure **1D** illustrates an exemplary network environment, in accordance with one embodiment.

15 Figure **2** shows a representative hardware environment associated with the computers of Figure **1D**.

Figure **3** illustrates a network analyzer framework that may be implemented in the context of the architecture of Figures **1D** and **2**.

Figure **4** shows a method for translating protocol decode objects, in accordance with one embodiment.

25 Figure **5** illustrates a graphical user interface for displaying textual identifiers associated with protocol decode objects, in accordance with one embodiment.

### DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figure 1D illustrates a network architecture 100, in accordance with one  
5 embodiment. As shown, a plurality of networks 102 is provided. In the context of  
the present network architecture 100, the networks 102 may each take any form  
including, but not limited to a local area network (LAN), a wide area network  
(WAN) such as the Internet, etc.

10 Coupled to the networks 102 are data servers 104 which are capable of  
communicating over the networks 102. Also coupled to the networks 102 and the  
data servers 104 is a plurality of end user computers 106. In the context of the  
present description, such end user computers 106 may include a web server, desktop  
computer, lap-top computer, hand-held computer, printer or any other type of  
15 hardware/software.

In order to facilitate communication among the networks 102, at least one  
gateway 108 is coupled therebetween. It should be noted that each of the foregoing  
network devices as well as any other unillustrated devices may be interconnected by  
20 way of a plurality of network segments. In the context of the present description, a  
network segment includes any portion of any particular network capable of  
connecting different portions and/or components of a network.

Figure 2 shows a representative hardware environment that may be  
25 associated with the data servers 104 and/or end user computers 106 of Figure 1D, in  
accordance with one embodiment. Such figure illustrates a typical hardware  
configuration of a workstation in accordance with a preferred embodiment having a  
central processing unit 210, such as a microprocessor, and a number of other units  
interconnected via a system bus 212.

30

The workstation shown in Figure 2 includes a Random Access Memory (RAM) 214, Read Only Memory (ROM) 216, an I/O adapter 218 for connecting peripheral devices such as disk storage units 220 to the bus 212, a user interface adapter 222 for connecting a keyboard 224, a mouse 226, a speaker 228, a microphone 232, and/or other user interface devices such as a touch screen (not shown) to the bus 212, communication adapter 234 for connecting the workstation to a communication network 235 (e.g., a data processing network) and a display adapter 236 for connecting the bus 212 to a display device 238.

The workstation may have resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. It will be appreciated that a preferred embodiment may also be implemented on platforms and operating systems other than those mentioned. A preferred embodiment may be written using JAVA, C, and/or C++ language, or other programming languages, along with an object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to develop complex applications.

Figure 3 illustrates a network analyzer framework 300 that may be implemented in the context of the architecture of Figures 1D and 2. Of course, the network analyzer framework 300 may be implemented in any desired context.

As shown, a protocol decoder 302 is provided that is coupled to any desired device and/or network segment for performing network analysis. One example of such protocol decoder 302 is the Sniffer® device manufactured by Network Associates®, Inc. In use, the protocol decoder 302 is capable of decoding frames of network communication collected from the associated device and/or network segment in order to generate protocol decode objects [i.e. SNMP (ASN.1)]. In addition to decoding, the protocol decoder 302 may further be able to collect any additional information for the purpose of determining why network performance is



slow, understanding the specifics about excessive traffic, and/or gaining visibility into various parts of the network.

5 In the context of the present description, a frame may refer to any component (i.e. packet, a group of packets, etc.) of network communications. Further, protocol decode objects may refer to a buffer, memory, a table or any other set of data that is associated with a specific component of communication protocol. Decoding may refer to any process or method capable of generating the protocol decode objects.

10 In use, a plurality of frames is received by the protocol decoder 302. Next, the frames are decoded in order to generate protocol decode objects each with a numerical identifier associated therewith. Next, the numerical identifier is translated to a textual identifier. The textual identifiers associated with the protocol decode objects are then displayed for facilitating the use of the protocol decode objects  
15 during network analysis. As an option, the textual identifiers may include alphanumeric text descriptive of the protocol decode objects.

20 In order to accomplish this, the numerical identifier is translated to a textual identifier utilizing a map stored in a database 304. To generate such map, a list of management information bases (MIBs) 306 is initially received from a user. Table #1 illustrates an exemplary MIB.

Table #1

25 RFC1213-MIB DEFINITIONS ::= BEGIN  
  
IMPORTS  
    mgmt, NetworkAddress, IpAddress, Counter, Gauge,  
        TimeTicks  
30 FROM RFC1155-SMI  
OBJECT-TYPE  
    FROM RFC-1212;  
  
-- This MIB module uses the extended OBJECT-TYPE macro as  
35 -- defined in [14];

4009000001.030102

```
-- MIB-II (same prefix as MIB-I)

5      mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }

-- textual conventions

10     DisplayString ::=
      OCTET STRING
-- This data type is used to model textual information taken
-- from the NVT ASCII character set.  By convention, objects
-- with this syntax are declared as having
--
15     SIZE (0..255)

PhysAddress ::=
      OCTET STRING
-- This data type is used to model media addresses.  For many
20 -- types of media, this will be in a binary representation.
-- For example, an ethernet address would be represented as
-- a string of 6 octets.

25     -- groups in MIB-II

      system      OBJECT IDENTIFIER ::= { mib-2 1 }

      interfaces  OBJECT IDENTIFIER ::= { mib-2 2 }

30     at          OBJECT IDENTIFIER ::= { mib-2 3 }

      ip          OBJECT IDENTIFIER ::= { mib-2 4 }

35     icmp        OBJECT IDENTIFIER ::= { mib-2 5 }

      tcp         OBJECT IDENTIFIER ::= { mib-2 6 }

      udp         OBJECT IDENTIFIER ::= { mib-2 7 }

40     egp         OBJECT IDENTIFIER ::= { mib-2 8 }

-- historical (some say hysterical)
-- cmot          OBJECT IDENTIFIER ::= { mib-2 9 }

45     transmission OBJECT IDENTIFIER ::= { mib-2 10 }

      snmp        OBJECT IDENTIFIER ::= { mib-2 11 }

50     -- the System group
```

5 -- Implementation of the System group is mandatory for all  
-- systems. If an agent is not configured to have a value  
-- for any of these variables, a string of length 0 is  
-- returned.

10 sysDescr OBJECT-TYPE  
SYNTAX DisplayString (SIZE (0..255))  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
"A textual description of the entity. This value  
should include the full name and version  
identification of the system's hardware type,  
15 software operating-system, and networking  
software. It is mandatory that this only contain  
printable ASCII characters."  
::= { system 1 }

20 sysObjectID OBJECT-TYPE  
SYNTAX OBJECT IDENTIFIER  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
25 "The vendor's authoritative identification of the  
network management subsystem contained in the  
entity. This value is allocated within the SMI  
enterprises subtree (1.3.6.1.4.1) and provides an  
easy and unambiguous means for determining `what  
30 kind of box' is being managed. For example, if  
vendor `Flintstones, Inc.' was assigned the  
subtree 1.3.6.1.4.1.4242, it could assign the  
identifier 1.3.6.1.4.1.4242.1.1 to its `Fred  
Router'.  
35 ::= { system 2 }

40 sysUpTime OBJECT-TYPE  
SYNTAX TimeTicks  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
"The time (in hundredths of a second) since the  
network management portion of the system was last  
re-initialized."  
45 ::= { system 3 }

50 sysContact OBJECT-TYPE  
SYNTAX DisplayString (SIZE (0..255))  
ACCESS read-write  
STATUS mandatory  
DESCRIPTION

40090261.030402

"The textual identification of the contact person  
for this managed node, together with information  
on how to contact this person."

::= { system 4 }

5

sysName OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..255))

ACCESS read-write

STATUS mandatory

10

DESCRIPTION

"An administratively-assigned name for this  
managed node. By convention, this is the node's  
fully-qualified domain name."

::= { system 5 }

15

sysLocation OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..255))

ACCESS read-write

STATUS mandatory

20

DESCRIPTION

"The physical location of this node (e.g.,  
'telephone closet, 3rd floor')."

::= { system 6 }

25

sysServices OBJECT-TYPE

SYNTAX INTEGER (0..127)

ACCESS read-only

STATUS mandatory

30

DESCRIPTION

"A value which indicates the set of services that  
this entity primarily offers.

35

The value is a sum. This sum initially takes the  
value zero. Then, for each layer, L, in the range  
1 through 7, that this node performs transactions  
for, 2 raised to (L - 1) is added to the sum. For  
example, a node which performs primarily routing  
functions would have a value of 4 ( $2^{(3-1)}$ ). In  
contrast, a node which is a host offering  
application services would have a value of 72  
( $2^{(4-1)} + 2^{(7-1)}$ ). Note that in the context of  
the Internet suite of protocols, values should be  
calculated accordingly:

45

layer functionality

- 1 physical (e.g., repeaters)
- 2 datalink/subnetwork (e.g., bridges)
- 3 internet (e.g., IP gateways)
- 4 end-to-end (e.g., IP hosts)
- 7 applications (e.g., mail relays)

50

100900001.030102

For systems including OSI protocols, layers 5 and  
6 may also be counted."  
 ::= { system 7 }

- 5           The list of MIBs **306** is compiled in order to generate the map. Table #2  
illustrates an exemplary map.

Table #2

10

Numerical_Identifier1	Textual_Identifer1
Numerical_Identifier2	Textual_Identifer2
Numerical_Identifier3	Textual_Identifer3
Numerical_Identifier4	Textual_Identifer4
Numerical_Identifier5	Textual_Identifer5

15

- As shown, the map may include a list of the numerical identifiers each with  
an associated textual identifier. One example of a numerical identifier includes  
"1.3.6.1.2.1.16.14.1.1.11.16.280.4294967295.0." Further, an exemplary associated  
textual identifier includes "mib-2.rmon(16).nlHost(14).hlHostControlTable(1)  
20 .hlHostControlEntry(1). hlHostControlOwner(11).16.280.4294967295.0."

- The map of Table #2 is generated by using MIBs like that of Table #1 to  
automatically generate a textual identifier for every possible numerical identifier.  
This is accomplished by combining textual strings associated with specific numbers  
25 of particular components of the numerical identifier.

- In use, the database **304** associated with the protocol decoder **302** is loaded  
with the map so that the numerical identifier may be translated to the textual  
identifier utilizing the map during the decoding. As an option, the map may include  
30 a look-up table.

Figure 4 shows a method **400** for translating protocol decode objects, in  
accordance with one embodiment. The method **400** may be implemented in the

context of the architecture of Figure 4. Of course, however, the method 400 may be implemented in any desired environment.

Initially, in operation 402, a list of MIBs is received from a user. Such MIBs  
5 may be selected manually via a graphical user interface, or in any desired automatic manner. Such MIBs may be selected based on which components of a particular protocol are to be decoded. With reference again to Figure 1B, MIBs associated with various objects of the hierarchical tree structure (i.e. iso, org, internet, directory, management, etc.) may be selected.

10

Next, in operation 404, the list of MIBs is compiled in order to generate a map. The map may include any look-up table or any other type of data structure capable of indicating a correlation between numerical identifiers associated with the protocol decode objects and textual identifiers, in a manner that will soon be set  
15 forth. The manner in which the list of MIBs is compiled may be accomplished in any desired manner. For example, the map may be compiled in the manner set forth in Table #2 hereinabove.

A decoder may then be loaded with the map. See operation 406. This may  
20 be accomplished by loading the map in a database associated with the decoder. Of course, the decoder may be loaded with the map in any manner that makes the map accessible to the decoder.

Frames may then be decoded during process 407. In particular, a frame is  
25 received and decoded in operation 408. It should be noted that the frame may be received from a capture file stored previously, or the frame may be received in real-time. At least one protocol decode object is then generated based on the frame.

Each protocol decode object is equipped with a numerical identifier  
30 associated therewith during the decoding. This numerical identifier is identified in operation 410. The numerical identifier is then translated to a textual identifier

utilizing the map in operation **412**. This may be accomplished by way of a look-up operation, or any other desired translation procedure.

Once translated, the textual identifier associated with the protocol decode  
5 object is displayed in operation **414** for facilitating the use of the protocol decode object during network analysis. More information relating to an exemplary graphical user interface capable of displaying the textual identifier will be set forth during reference to Figure 5.

10 Next, it is determined in decision **416** whether another frame exists. If so, the various operations **408** – **414** are repeated for each frame.

Figure 5 illustrates a graphical user interface **500** for displaying textual  
15 identifiers associated with protocol decode objects, in accordance with one embodiment. As shown, associated with each object **502** is a textual identifier **504** which identifies the object resulting from the decoding in an improved manner.

The present technique may thus use a MIB compile operation in conjunction  
20 with protocol decodes. The MIB compile operation may compile all MIBs that the user wants to see in readable form and creates a database of mappings between a numeric representation of object identifiers and textual names of each object identifier. The protocol decoder may use this database to search for names using the numeric value of the object identifier as a key.

25 The present method is therefore flexible and any MIB can be added to/removed from the database without any changes in code. Any MIB with a valid syntax (including private MIBs) may be used.

While various embodiments have been described above, it should be  
30 understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of a preferred embodiment should not be

limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

40090001.0301.02